Государственное бюджетное общеобразовательное учреждение

Лицей №150

**ТЕМА ПРОЕКТА:**

Создание программы для хранения паролей.

**Тип проекта:**

Информационный

**Выполнила:**

Полякова Юлия Вадимовна, ученица 10 «А» класса.

**Руководитель:**

Мохов Дмитрий Васильевич

**Оглавление.**

1.Введение…………………………………………………………………………4

2. Цели и задачи…………………………………………………………………...5

3. Глава 1. Теоретический обзор

3.1. Объектно-ориентированный Паскаль…………...…………………...6

3.2. Lazarus.………………………………………………………..………..6

3.3. Шифр Виженера……………………………………………………….6

4. Глава 2. Практическая часть

4.1. Используемые ресурсы………………………………………………12

4.2. Подготовительный этап……………………………………………...12

4.3. Описание продукта…………………………………………………..12

5. Выводы………………………………………………………………………...13

6. Литературный материал………………………………………………………14

7.Приложения……………………………………………………………………15

8. Глоссарий……………………………………………………………………...22

**1.Введение.**

Сравнительно недавно в нашу жизнь пришли компьютеры. Вместе с ними пришло и до сих пор прибывает множество программ. Сегодня пользователь имеет учетные записи в компьютерных играх, социальных сетях, email разных электронных почт, а так же каждый работник имеет аккаунт в специальных программах, имеющих доступ к документам и данным места работы. Каждая учетная запись требует пароль, который может иметь свои требования. Например, пароль должен иметь буквы латинского алфавита, некоторое количество цифр, заглавных букв и специальных символов. Отдельные корпоративные программы требуют смены пароля через определенный отрезок времени для повышения уровня защиты данных.

Имея огромное количество паролей, пользователь сталкивается с такими проблемами как неверно введенный или вовсе забытый пароль, перепутанный login или email. Каждая проблема несет череду последствий, например, блокировкой данной учетной записи и созданием новой, что может повлечь потерю данных. Человеку свойственно что-либо забывать и он будет хранить данные для входа на бумажных носителях (в записной книжке, листочках и т.п.), что чревато возможностью кражи или потери данных.

Эта проблема актуальна сегодня для всех, так как подавляющее большинство людей всего мира использует средства защиты от кражи данных, поэтому я решила сама написать программу, позволяющую хранить и изменять пароли. Программа в свою очередь будет защищать данные для входа шифром Виженера, который взломать довольно трудно.

**2. Цели и задачи.**

**Цель:**

Написать программу в оболочке Lazarus к середине декабря для пользователей, использующих пароли различных программ. С социальной точки зрения данная программа облегчит работу пользователей с электронными ресурсами, которые требуют паролей, и избавит от затрат времени на решение проблем, связанных с паролем.

**Задачи:**

1. Изучить теоретическую базу для дальнейшего использования оболочки Lazarus.
2. Написание в данной оболочке программы для сохранения многочисленных паролей, которые могут понадобиться пользователю для работы с ПК.
3. Применение готовой программы на практике в ООО «КИРИШИАВТОС**ЕРВИ**С»
4. Сделать выводы

**Актуальность:**

Данная работа позволяет безопасно хранить пароли в зашифрованном виде в специально разработанной программе

**Проблема:**

Как написать программу для хранения неограниченного количества паролей для разных программ и сервисов для 1-го пользователя, используя среду разработки Lazarus.

**Противоречие:**

Буфер обмена Windows может запомнить единовременно один пароль, а нужно оперативно переключаться между разными паролями для разных программ (любая база данных ограничена в работе в фоновом режиме и после превышения ограничения автоматически закрывается, требует нового ввода пароля)

**3. Глава 1 – Теоретический обзор.**

**3.1. Объектно-ориентированный Паскаль.**

**Объектно-ориентированное программирование – методология, основанная на представлении программы в виде совокупности**[**объектов**](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))**, каждый из которых является экземпляром определённого**[**класса**](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))**, а классы образуют иерархию наследования. Идеологически, такой вид программирования нацелен на моделирование информационных процессов, т.е. совершенствование выполнения основной задачи структурного программирования. Задачей данного вида программирования является структурирование информации с точки зрения управляемости, что особенно облегчает процесс реализации крупных проектов.**

**Управляемость для иерархических систем предполагает уменьшение избыточности данных и их целостность, что будет легче для понимания управления. Таким образом, выполняется одна из задач программиста: трансляция понимания задачи в наиболее удобную форму. Основные принципы объектно-ориентированного программирования:**

1. **Абстракция – придание объекту характеристик, отличающих его от других объектов, четко определяя концептуальные границы. Основная задача состоит в том, чтобы отделить способ использование составных объектов данных от деталей их реализации в виде более простых объектов. Такой подход является основой, что позволяет работать с объектами, не вдаваясь в особенности их реализации.**
2. **Инкапсуляция – свойство языка, позволяющее пользователю взаимодействовать с программным компонентом посредством предоставляемого интерфейса и защитить необходимые для его работы данные. Один из четырех важнейших механизмов объектно-ориентированного программирования наряду с абстракцией, наследованием и полиморфизмом.**
3. **Наследование – механизм, позволяющий описать новый класс на основе существующего (родительского), при этом свойства нового класса заимствуются. Этот механизм запускается для того, чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя все остальные данные, учтенные на предыдущем «шаге».**
4. **Полиморфизм – механизм для определения точки, в которой единое управление лучше разделить на параллельные потоки или наоборот, собрать в одно целое.**

**Object Pascal (объектно-ориентированный Па**скаль) -  **это надмножество (дальнейшее развитие) языка Pascal, в который введено понятие объекта и классов объектов. Таким образом, реализована концепция объектного ориентирования. В дальнейшем будет использоваться аббревиатура. Object Pascal был разработан фирмой**  **[Apple Computer](https://ru.wikipedia.org/wiki/Apple_Computer" \o "Apple Computer) в**[**1986 году**](https://ru.wikipedia.org/wiki/1986_%D0%B3%D0%BE%D0%B4)**группой**[**Ларри Теслера**](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%80%D1%80%D0%B8_%D0%A2%D0%B5%D1%81%D0%BB%D0%B5%D1%80)**, который консультировался с [Никлаусом Виртом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%BA%D0%BB%D0%B0%D1%83%D1%81_%D0%92%D0%B8%D1%80%D1%82" \o "Никлаус Вирт). Ранее Паскаль назывался**  **[Clascal](https://ru.wikipedia.org/w/index.php?title=Clascal&action=edit&redlink=1" \o "Clascal (страница отсутствует)) и был доступен только на компьютере Apple Lisa. Основными реализациями данного языка программирования являются: Delfi, Oxygene, Free Pasсal, Visual Pasсal. Рассмотрим преимущества и недостатки Object Pascal.**

1. **Преимущества:**
   1. **Доступные IDE (Integrated Development Environment – интегрированная среда разработки – система программных средств для разработки программных обеспечений).**

**IDE объединяет несколько инструментов, специально предназначенных для разработки. Эти инструменты обычно включают редактор, предназначенный для работы с кодом (например, подсветка синтаксиса и автодополнение); инструменты сборки, выполнения и отладки; и определённую форму системы управления версиями.**

* 1. **Понятный синтаксис**
  2. **Минимальные затраты времени на освоение  IDE и создание прикладных приложений.**

1. **Недостатки:**
   1. **Малое количество библиотек по сравнению с программами семейства  C.**
   2. **Недостаточная гибкость, т.к. эволюционировал от начального Pascal.**
   3. **Слабая поддержка со стороны разработчиков IDE.**
   4. **Ограничение функций.**

**Object Pascal направлен на создание простых приложений, не требующих лишних или малоиспользуемых дополнений. С помощью Object Pascal были написаны Skype, The Bat!**

**3.2. Lazarus.**

**Lazarus – открытая среда разработки программного обеспечения на языке Object Pascal для компилятора Free Pascal, так же данная среда интегрирована для кроссплатформенной разработки приложений в Delphi-подобном окружении. Данная среда поддерживает для компиляции такие операционные системы как: Lunux, Mac OS, Windows, Android, ReactOS. Lazarus основан на библиотеке виртуальных компонентов Lazarus Component Library.**

**Функции:**

1. Поддерживает преобразование проектов Delphi;
2. Реализован основной набор элементов управления;
3. Редактор форм и инспектор объемов максимально приближены к Delphi;
4. Присутствует интерфейс отладки с внешним отладчиком GDB;
5. Полностью юникодный (UTF-8) интерфейс и редактор, что предотвращает проблемы с портированием кода с национальными символами;
6. Имеет собственный формат управления пакетами
7. Автосборка самого себя «в один клик»
8. Поддержка всех синтаксисов Pascal
9. Наличие редактора с системой подсказок, гипертекстовую навигацию по исходным текстам и авторазрешения.

В среде Lazarus программист получает возможность не только создавать программный код, но и наглядно показывать системе, что конкретно хотел бы увидеть разработчик программы. Технология визуального программирования позволяет буквально строить интерфейс будущей программы из специальных компонентов, несущие нужные функции. Количество компонентов в данной среде максимально велико и каждый такой компонент содержит уже готовый программный код и необходимые данные, что упрощает задачу программиста и укорачивает в несколько раз программный код, что так же удобно при поиске ошибок и для визуального восприятия написанного кода. Наличие готовых компонентов, гипертекстовая навигация и система подсказок значительно сокращают время и облегчают процесс написания программы.

На рисунке №1 представлен интерфейс среды Lazarus.

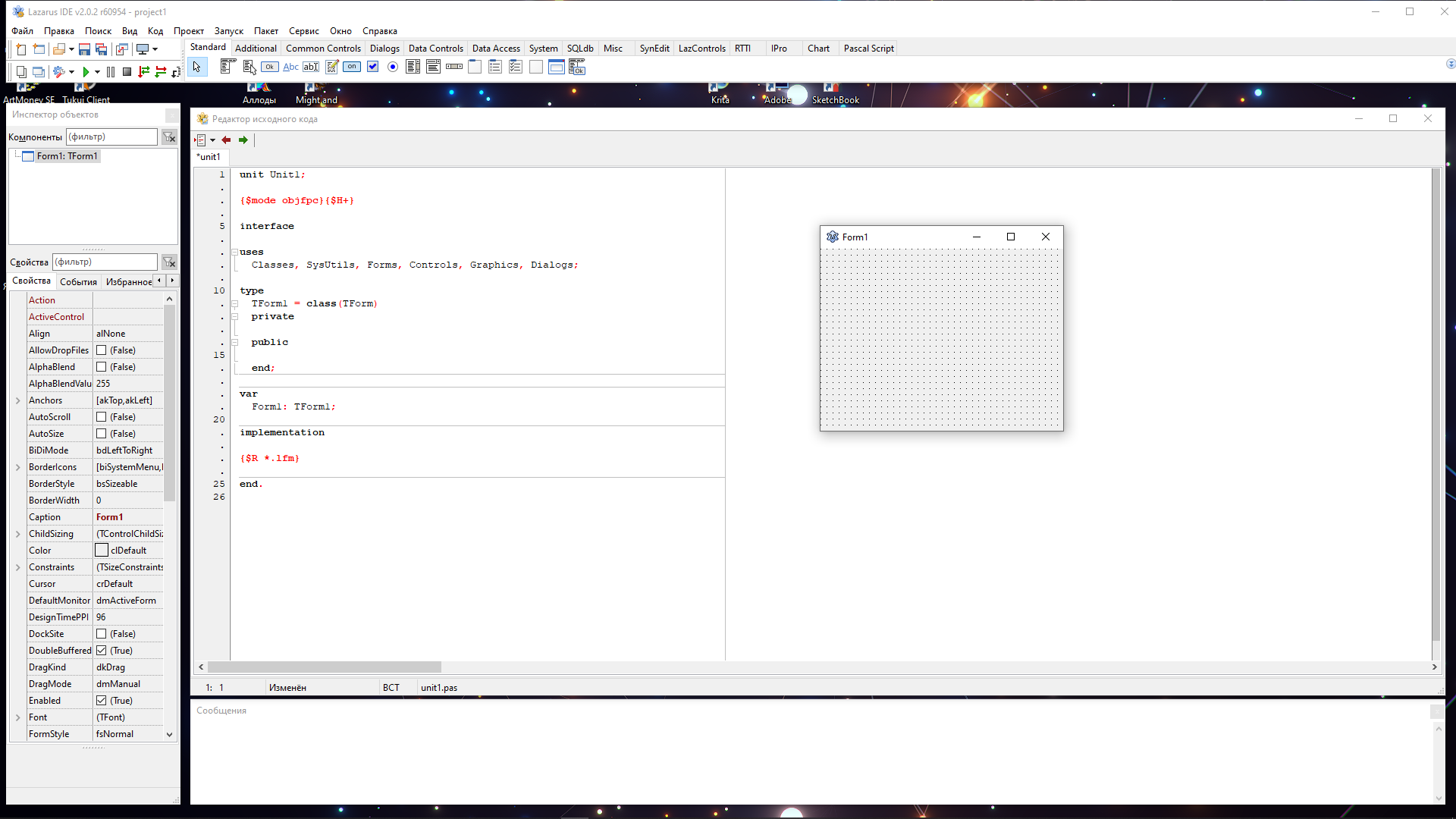


Рисунок 1. Интерфейс IDE Lazarus.

**3.3. Шифр Виженера.**

Шифр Виженера – метод полиалфавитного шифрования буквенного текста с использованием кодового слова. Этот шифр прост для понимания и реализации, но на протяжении трех столетий он не поддавался взлому. По этой причине он получил название «неразгаданный шифр». Этот метод является простой формой многоалфавитной замены.

Впервые Шифр Виженера описал Джовани Баттиста Белласо, однако по ошибке был приписан другому имени, Блезу Виженеру, французскому дипломату. В 1518 году в развитии криптографии был сделан новый шаг. Аббат Иоганнес Тритемий, настоятель монастыря в Вюрцбурге, написал книгу «Полиграфия», в которой описывается ряд шифров. Один из них использует «[таблицу Тритемия](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0_%D0%A2%D1%80%D0%B8%D1%82%D0%B5%D0%BC%D0%B8%D1%8F&action=edit&redlink=1)» (более известна как «таблица Виженера») и развивает идею многоалфавитной замены. Система шифрования следующая: первая буква исходного текста шифруется по первой строке, вторая по второй и т.д. После использования последней строки следующая буква снова шифруется по первой строке. В шифре Тритемия отсутствует ключ, секретом является сам способ шифрования.

Далее за развитие криптографии принялся итальянец Джованни Белазо. В 1553 году он выпустил брошюру «Шифр синьора Белазо». В этом шифре ключом является фраза или слово. Пароль записывался периодически над буквами открытого текста. Буква пароля, стоящая над соответствующей буквой открытого текста, указывала номер строки в таблице Тритемия, по которой следует проводить замену (шифрование) это буквы.

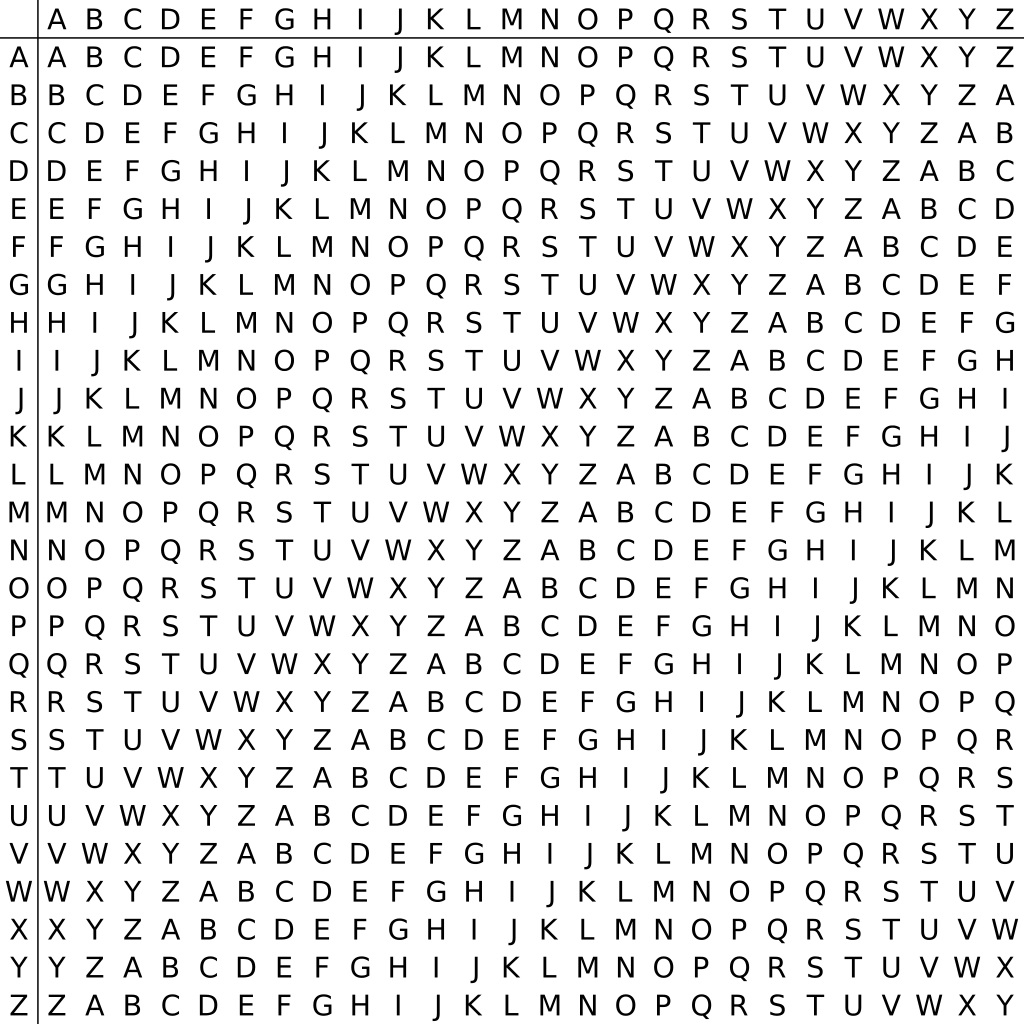


Рисунок 2. Квадрат Виженера (таблица Виженера)

Следующим составляющим звеном в нынешнем шифре Виженера стал соотечественник Белазо [Джованни Батиста Порта](https://ru.wikipedia.org/wiki/%D0%94%D0%B6%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B8_%D0%91%D0%B0%D1%82%D1%82%D0%B8%D1%81%D1%82%D0%B0_%D0%9F%D0%BE%D1%80%D1%82%D0%B0). Он предложил отказаться от алфавитного порядка следования букв в первой строке таблицы Тритемия и заменить этот порядок на некоторый произвольный, являющийся ключом шифра. Строки таблицы по-прежнему циклически сдвигались. В своей книге «О тайной переписке», (вышедшей в 1563 году) Порта предложил [биграммный шифр](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80" \o "Биграммный шифр), а также привел описание механического дискового устройства, реализующего биграммную замену.

Далее в середине XVI века в Италии появляется книга [Дж. Кардано](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%B4%D0%B0%D0%BD%D0%BE,_%D0%94%D0%B6%D0%B5%D1%80%D0%BE%D0%BB%D0%B0%D0%BC%D0%BE) «О тонкостях» с дополнением «О разных вещах». Там нашли отражение новые идеи криптографии: использование части самого передаваемого открытого текста в качестве ключа шифра (идея «самоключа») и новый способ шифрования, который вошел в историю как «[решетка Кардано](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D1%91%D1%82%D0%BA%D0%B0_%D0%9A%D0%B0%D1%80%D0%B4%D0%B0%D0%BD%D0%BE)».

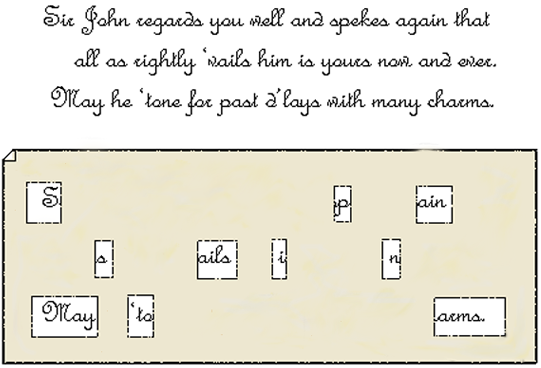


Рисунок 3. Записка, зашифрованная шифром Кардано.

 На рисунке №3 представлен сам шифр Кардано. Записка гласит: «Сэр Джон высоко ценит Вас и снова повторяет, что все, что доступно ему, теперь ваше, навсегда. Может ли он заслужить прощение за свои прежние промедления посредством своего обаяния», шифрованное послание: «В мае Испания направит свои корабли на войну».

Посол Франции в Риме [Блез де Виженер](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80,_%D0%91%D0%BB%D0%B5%D0%B7" \o "Виженер, Блез), познакомившись с трудами Тритемия, Белазо, Кардано, Порта, Альберти, также увлекся криптографией. В 1585 году он написал «Трактат о шифрах», в котором излагаются основы криптографии.

По смыслу, Шифр Виженера состоит из нескольких шифров Цезаря (в данном шифре каждая буква сдвигается на несколько позиций).

Например:

* Исходный текст: «В кабинете информатики много компьютеров»
* Ключ: «Лимон»
* Полученный шифр: «н умпцщняу цщэыяълыхщц шцысь цчщюййысяьн»

То есть, шифрование происходит по формуле cj =(mj+kj)\*mod n

Расшифровка происходит при помощи той же таблицы Виженера и формул. cj =(mj+n-kj)\*mod n

В данных формулах n – количество букв в алфавите, mj  - буквы открытого текста, kj – буквы ключа.

**4. Глава 2 – Практическая часть.**

В работе использовалась бесплатная программа-среда разработки ПО Lazarus. Ver. 2.0.2

**4.1.Используемые ресурсы:**

* Компьютер на базе Intel Pentium
* Оболочка для написания программных обеспечений Lazarus
* Обучающая литература и пособия по программированию на языке **Pascal**

**4.2. Подготовительный этап.**

В начале работы над практической частью мы определили, какие команды нужно «исполнять» программе, и составили блок-схему (приложение 1), которая исполнила роль плана написания программы.

**4.3. Описание продукта.**

function Tfr.deshifr(s: string): string; //функция дешифрования

var

i: integer;

st: string;

begin

for i:=1 to Length(s) do

if (s[i]<> '') and (s[i]<> '.') and (s[i]<> ';') and (s[i]<> ',') then

s[i]:= Char((Ord(s[i]) - Ord(k[(i mod Length(k)) + 1])) mod 256);

st := Copy(s, 1, Length(s) - l);

Result:= st;

end;

Рисунок №4 .Часть кода программы. Функция дешифрования.

На рисунке 4 представлена часть кода программы, функция дешифрования. Полностью код нигде нельзя увидеть, так как это личная разработка.

Продукт проекта – это программа, запоминающая пароли, написанная в среде разработки ПО Lazarus на языке Pascal. Простая в плане интерфейса и понимания содержимого, но сложная в плане защиты данных. Все наименования надежно защищены, так как алгоритм шифрования вычислить невозможно. (приложение 4)

Такая программа актуальна в наши дни, так как многие пользуются ресурсами, требующими авторизации (это не только электронная почта, социальные сети и различные игровые ресурсы, но и такие «рабочие» ресурсы, как Ариадна, SAP, Directum (система электронного документооборота)). По правилам техники безопасности, пароли не должны повторяться, содержать латинские буквы, цифры и специальные символы, также раз в определенный срок их нужно менять. Из-за огромного потока информации пользователю довольно сложно запомнить все данные для авторизации. Данная программа может решить проблему с путаницей паролей. Нужно лишь нажать на нужный ресурс, вернуться к окну авторизации и одновременно нажать клавиши «ctrl» и «C».  
 Подобный продукт создать вовсе не тяжело, но трудоемко. Мне понадобилось время, чтобы изучить теоретический материал, помогающий изучить язык программирования, саму среду разработки ПО.

**5.Выводы.**

В результате проделанной работы

1. Был подобран литературный материал для реферативной части

2. Был освоен язык Pascal и навыки программирования на нет, а также был освоен шифр Виженера

3. При помощи IDE Lazarus можно создать безопасную программу, хранящую пароли от разных ресурсов, требующих авторизации.

**6.Список литературы.**

1. «Самоучитель по программированию на Free Pascal и Lazarus» Издательство УНИТЕХ, 2009, г. Донецк, Алексеев Е.Р., Чеснокова О.В., Кучер Т.В.
2. [Наглядная инструкция шифрования с помощью шифра Виженера](https://ru.wikihow.com/%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D1%82%D1%8C-%D0%B8-%D1%80%D0%B0%D1%81%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D1%8B%D0%B2%D0%B0%D1%82%D1%8C-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-%D1%88%D0%B8%D1%84%D1%80%D0%B0-%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B0)
3. [Основные сведения о шифре Виженера](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B0)
4. [Цикл лекций Ачкасова Вячеслава Юрьевича по программированию в Lazarus](http://intuit.valrkl.ru/course-1265/index.html#ID.23.section.1)
5. [Электронная книга Мансурова К. Т. Основы программирования в среде Lazarus 2010](http://www.freepascal.ru/download/pdf/osnovy_programmirovanija_v_srede_lazarus.pdf)

**7.Приложения.**

**Приложение 1. Блок-схема работы программы.**

Начало

Ввод данных: название ресурса, пароль

Сохранение данных

Добавление пункта с данными

Ожидание выбора пункта с данными

Выбор пункта с данными

Помещение данных в буфер обмена Windows

Помещение данных в буфер обмена Windows

ние данных в бу Помещение данных в буфер обмена Windows

фер обмена Windows

Помещение данных в буфер обмена Windows

Помещение данных в буфер обмена Windows

Помещение данных в буфер обмена Windows

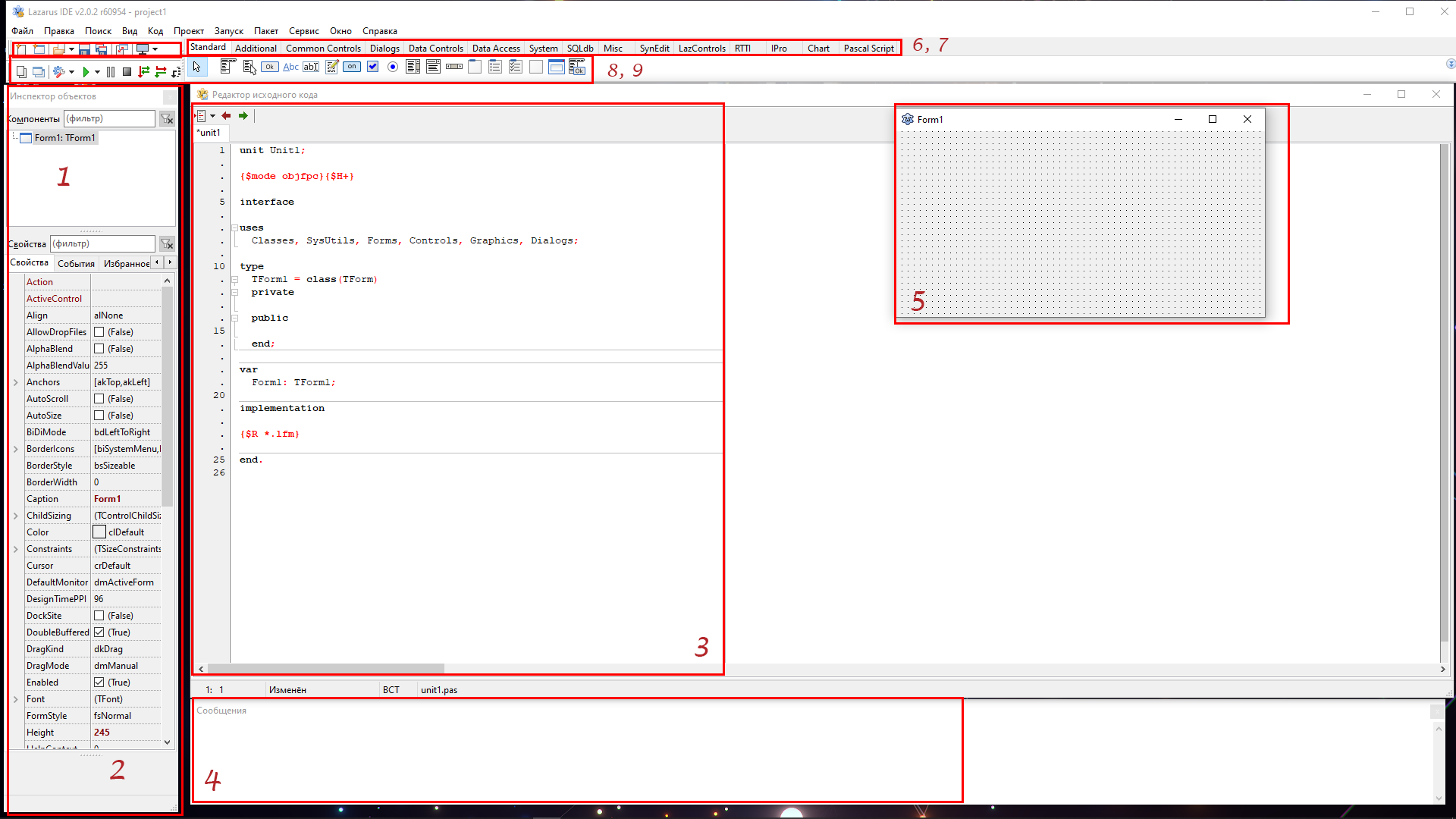
Помещение данных в буфер обмена Windows

Помещение данных в буфер обмена Windows

Помещение данных в буфер обмена Windows

Конец

**Приложение 2. Основные части IDE Lazarus.**



1. Окно созданных объектов, которые будут отображены в окне формы, т.е. объектов, которые будут видны в готовой программе
2. Окно свойств и событий1 выбранного объекта
3. Окно редактора кода, т.е. рабочая зона, где пишется сам код
4. Окно событий, т.е. окно, сообщающее о наличии ошибок и завершении проверки на наличие ошибок, предупреждающее о неиспользованных переменных
5. Окно формы, т.е. модель будущего результата
6. Панель быстрой работы с файлом, т.е. сохранение текущего документа, создание новых форм и пр.
7. Панель разделов готовых объектов
8. Панель работы с «активной» формой, т.е. демонстрация готовой части желаемого результата для осознания, получившегося и дальнейшего редактирования при необходимости
9. Панель готовых объектов

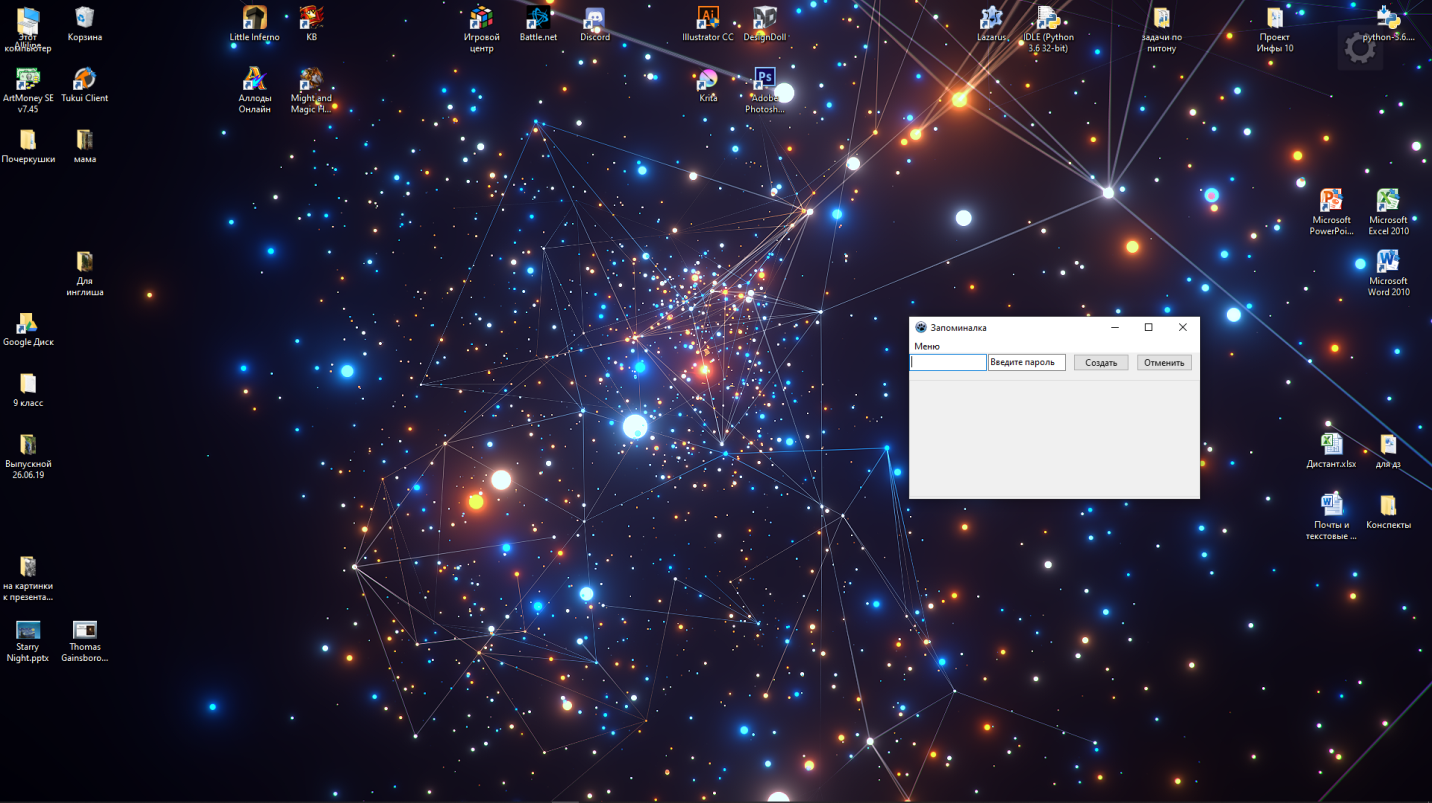
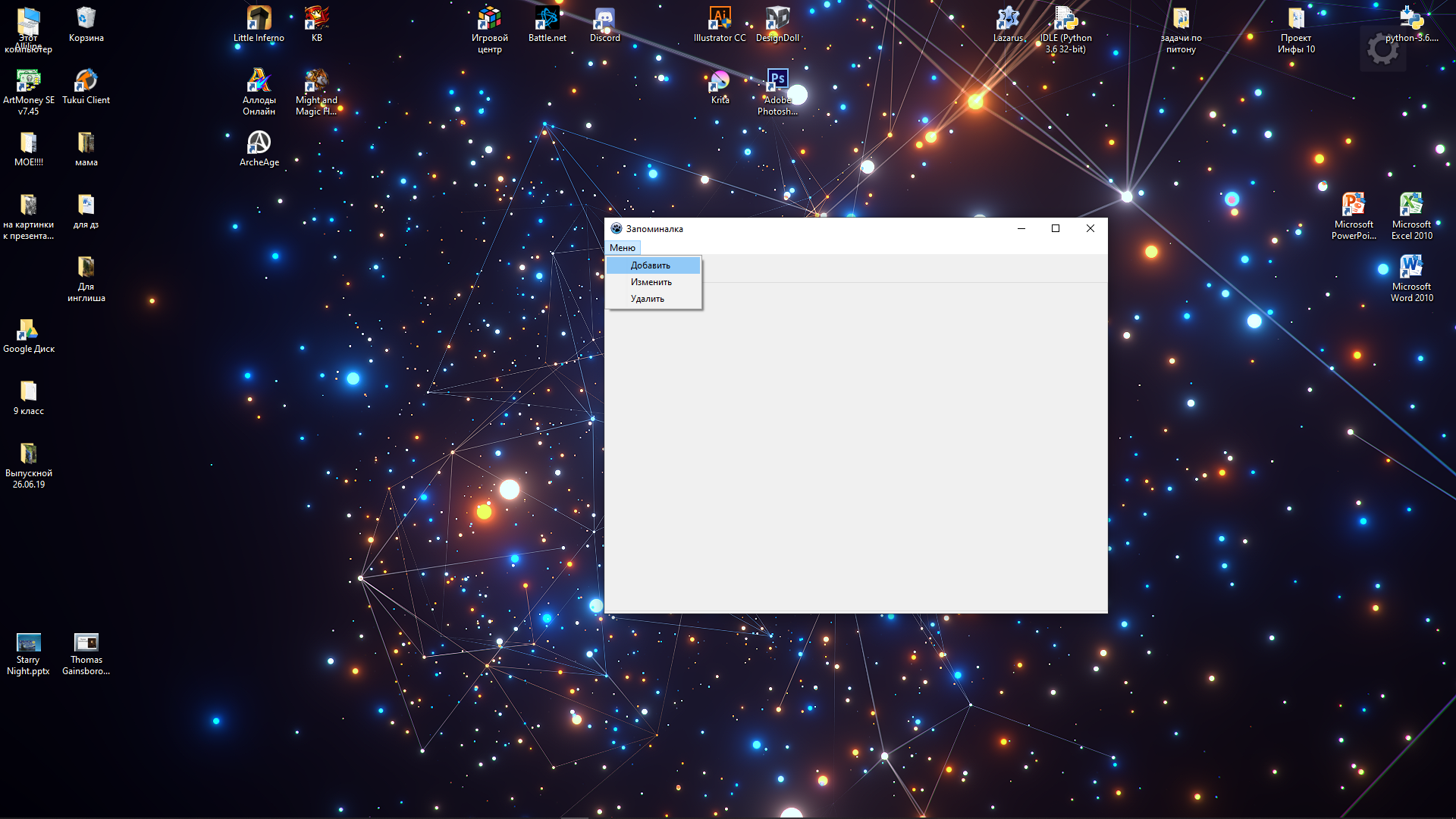
...Окно свойств и событий1 … - каждый объект имеет свои свойства и «События». В окне «Свойства» можно, например, задать имя объекту. В окне «События» задается действие, как следствие другому действию. Например, событие «OnClose» влечет закрытие всей формы.

**Приложение 3. Таблица свойств форм IDE Lazarus.**

Каждый объект имеет множество свойств, рассмотрим основные свойства формы и заключим их в таблицу:

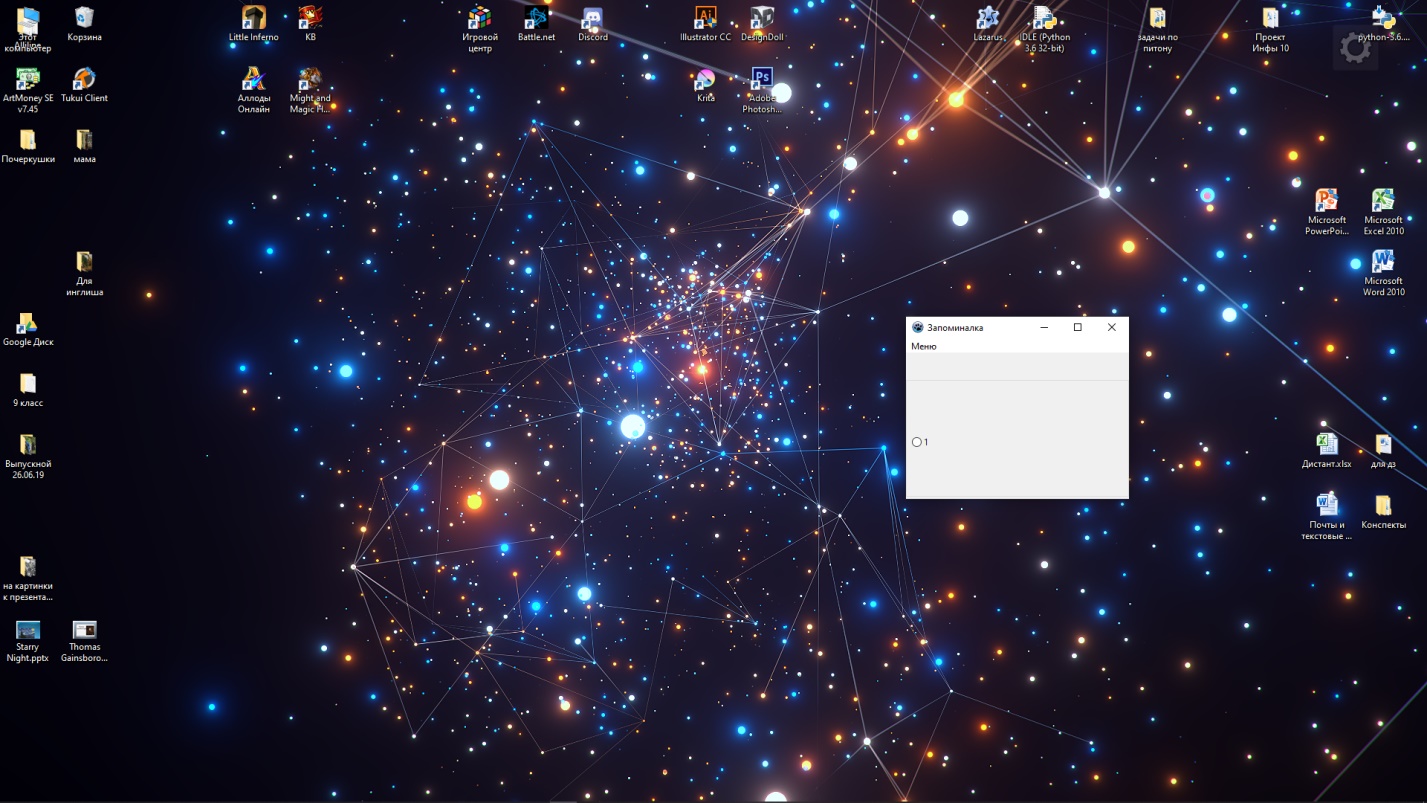
|  |  |
| --- | --- |
| *Свойства* | *Описание* |
| *Name* | *Имя компонента (формы)* |
| *Caption* | *Строковое свойство, которое отвечает за заголовок окна.* |
| *Align* | *Выравнивание компонента относительно окна:*  *alNone – нет выравнивания (как нарисовал, так и будет)*  *alBottom – выравнивание по нижнему краю.*  *alLeft – выравнивание по левому краю.*  *AlRight – выравнивание по правому краю.*  *AlTop – выравнивание по верхнему краю.* |
| *Width* | *Ширина окна.* |
| *Height* | *Высота формы.* |
| *BorderStyle* | *Вид границы (отвечает за вид оборки окна). Это свойство может принимать следующие значения.*  *bsSizeable – установлено по умолчанию. Стандартное окно, с нормальной оборкой, которое может изменять свои размеры.*  *bsDialog – окно выглядит в виде диалога.*  *bsNone – окно вообще без оборки.*  *bsSingle – окно с фиксированным размером и изменять его мышкой нельзя. Изменить размер можно только кнопкой Maximize.*  *bsSizeToolWin – окно с тонкой оборкой. Особенно это заметно в заголовке окна.* |
|  | *Свойство определяющее, какие кнопки должны присутствовать у окна:*  *biSystemMenu – показать меню (иконка слева в строке заголовка окна) и другие кнопки заголовка окна.*  *biMinimize – кнопка минимизации окна.*  *biMaximize – кнопка максимизации окна.*  *biHelp – кнопка помощи.* |
| *BorderIcons* |
| *Icon* | *Значок в заголовке диалогового окна, обозначающий кнопку вывода системного меню.* |
| *Color* | *Цвет клиентской области окна.* |
| *Font* | *Шрифт. Шрифт, используемый «по умолчанию» компонентами, находящимися на поверхности формы. Изменение свойства Font формы приводит к автоматическому изменению свойства Font компонента, располагающегося на поверхности формы. То есть компоненты наследуют свойство Font от формы (имеется возможность запретить наследование).* |
| *WindowState* | *Состояние окна после запуска.* |
| *Visible* | *Если оно равно true, то форма/компонент видим. Иначе форма/компонент невидим.* |
| *Position* | *Позиция окна при старте приложения.* |
| *Hint* | *Текст подсказки, который будет появляться в строке состояния при наведении мышкой на форму/компонент.* |
| *Enabled* | *Тип свойства – логический. Доступность компонента. Если это свойство равно true, то пользователь может работать с этим компонентом. Иначе компонент недоступен и окрашен серым цветом.* |
| *BorderWidth* | *Ширина оборки окна.* |
| *AutoSize* | *Определяет, должны ли компоненты на форме автоматически корректировать размеры формы.* |
| *AlphaBlend* | *Означает, имеет ли форма прозрачность. Если это свойство равно true, то окно будет прозрачным.* |

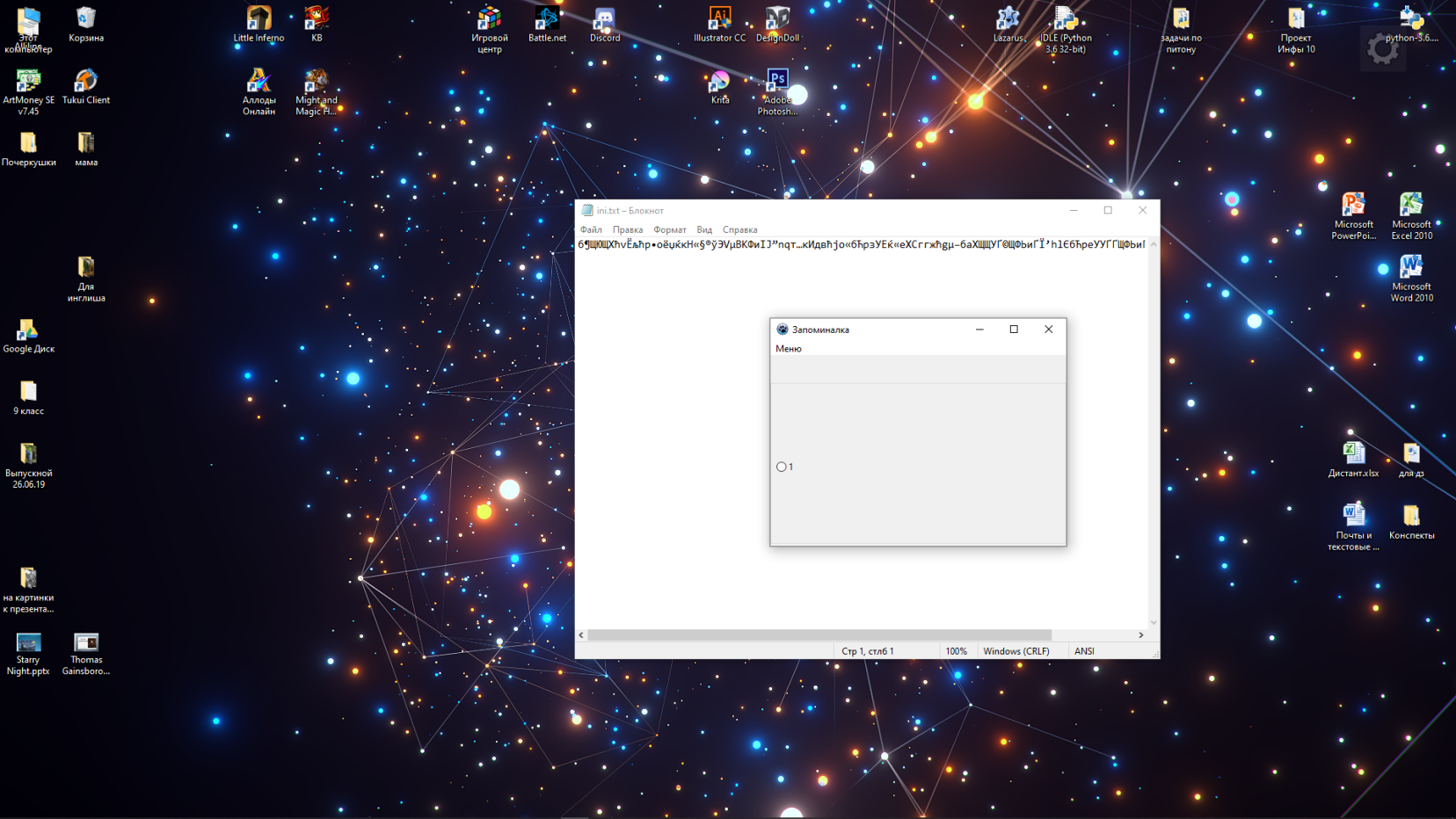
**Приложение 4. Описание продукта проекта.**

****

Скриншот 2. Создание нового пункта

Скриншот 1. Общее меню действий



****

Скриншот 4. Хранение данных

Скриншот 3. Готовый пункт

На скриншотах 1-3 представлена работа программы, на скриншоте 4 – где и как хранятся сохраненные данные. Важно обратить внимание на скриншоты 3 и 4.  
 Рассмотрим скриншот 3. Уже существует некий ресурс с названием «1» и для него n-ный пароль. Чтобы выбрать данный ресурс, достаточно просто нажать на область выбора, не дожидаясь каких-либо опознавательных знаков (галочки, крестика, подсветки и т.п.). Код написан так, что «сигнальных» знаков видно не будет, что может сбить с толку того, кто хочет авторизоваться под чьими-либо учетными данными. Владельца ПК же это не должно беспокоить, так как в случае нажатия на пункт пароль автоматически копируется в буфер обмена, а в случае возникновения необходимости изменить или удалить данные, программа попросит пользователя выбрать пункт с нужными данными.

Рассмотрим скриншот 4. На нем можно увидеть окно программы и текстовый документ с набором букв и символов, не несущим смысловой нагрузки с точки зрения русского языка. Все данные о рабочем окне программы (расположение на мониторе, высота и ширина окна), а также хранящиеся данные, находятся именно в этом текстовом документе в зашифрованном виде. Расшифровать такую запись, не зная слова-ключа, невозможно, так как шифрование строится на шифровании существующего текста шифром Виженера и добавлением определенного количества дополнительных символов, что повышает уровень безопасности. Также программа может запомнить свое местоположение на экране и размеры, которые пользователь может менять на свое усмотрение.

Скриншот 4. Хранение данных

**9.Глоссарий.**

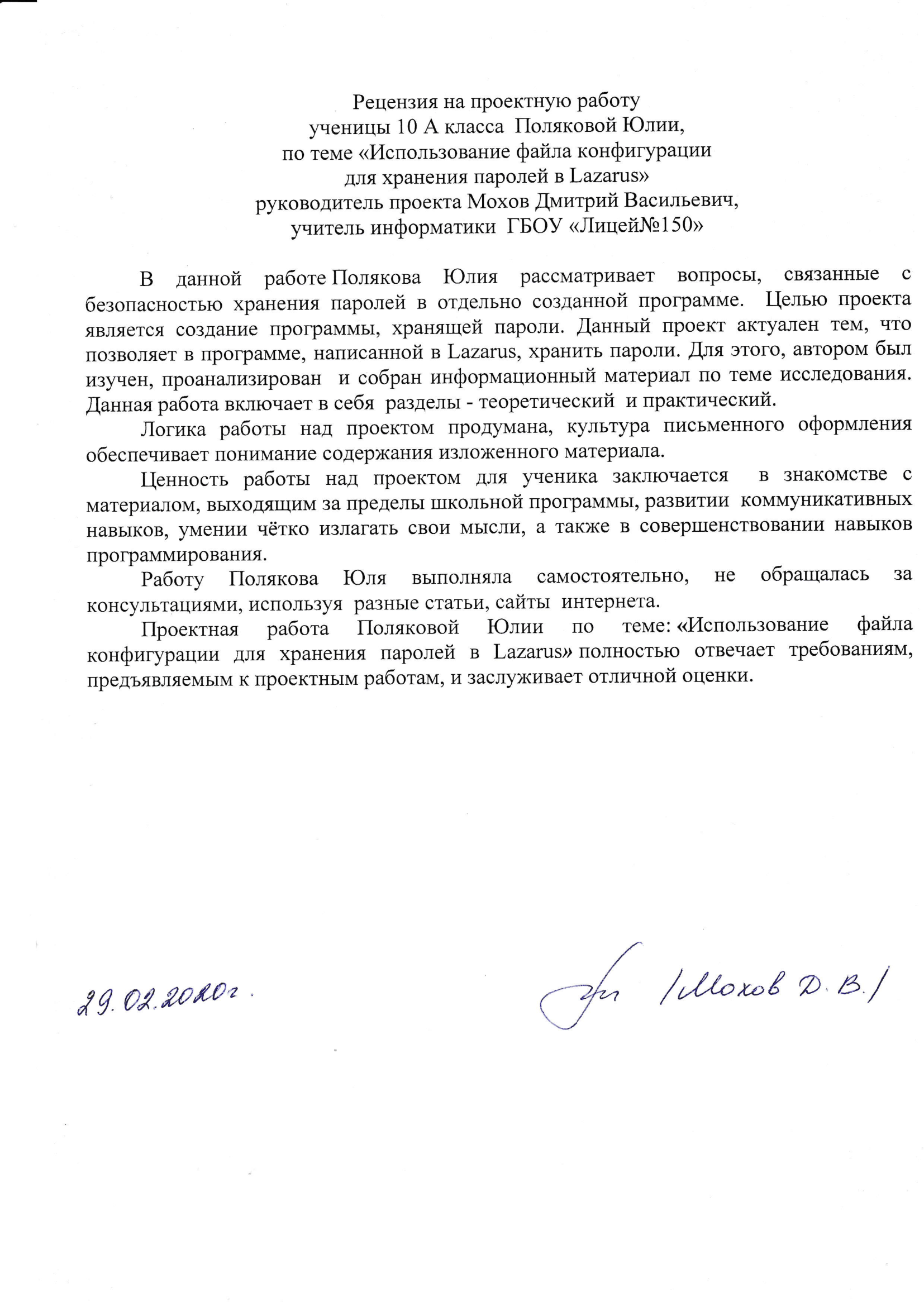
**GDB отладчик – GNU Debugger — переносимый отладчик проекта GNU, который работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования, включая Си, C++, Free Pascal, FreeBASIC, Ada, Фортран и Rust.**

**Integrated Development Environment – интегрированная среда разработки – система программных средств для разработки программных обеспечений.**

**Кроссплатформенная разработка — разработка со способностью программного обеспечения работать с двумя и более аппаратными платформами и операционными системами.**

**Методология — учение о методах, способах и стратегиях исследования предмета.**

**Структурное программирование – модель программирования, в основе которой лежит представление программы в виде иерархической структуры блоков.**

****